Mathematical Modeling Comparing Single vs. Collective Cell Migration in Breast Cancer Cells

Undergraduate Research Thesis

Project Advisor: Samir Ghadiali, Ph.D.

Jill Sanders

Department of Biomedical Engineering

College of Engineering

The Ohio State University

Columbus, OH

16th May 2012

Abstract

Cell migration is a key feature in metastatic breast cancer. Detachment from the primary tumor to other tissue sites makes breast cancer much more difficult to treat since the disease is no longer localized to its origin. A proposed method of cell detachment suggests cells move as a single unit rather than as a collective sheet. This theory is known as an epithelial to mesenchymal transition (EMT), which indicates a phenotype change allowing cells to become more invasive for increased tumorgenesis. However, this theory is not widely accepted, and little research quantifies or models this method of transition. This project attempts to provide predictive power to characterize single or collective cell migration in terms of mathematical modeling. Modeling was based off analysis from a series of wound healing assays comparing invasive breast cancer cells to a knockdown cell type. Three models were developed comparing a target cell's motility to its neighboring cells on both sides of the wound. Each model quantifies a combination of direction, distance, and time when comparing a target and neighbor cell's movement. Results from each model show the knockdown cells as being more correlated, implying that the cells move as a collective unit. These model results along with the results characterizing average velocity, distance, and directionality are indicative of an epithelial phenotype. The consistency of the results across each model support the notion that silencing gene expression in invasive breast cancer cells could change a mesenchymal phenotype to an epithelial phenotype, or MET. Overall the results of each model and parameter analysis provide a more in-depth analysis distinguishing single vs. collective cell migration in breast cancer cells. Furthermore, these models can be easily manipulated based on user preference and used to analyze cell migration in other cell lines.

Acknowledgements

I would like to thank Dr. Ghadiali for his time, help, and guidance throughout the course of this project. Without his support and the opportunity to work in the lab, this project would not have been possible. I would also like to acknowledge Dr. Kniss for serving on my committee and his lab members for their extensive research in breast cancer cells. One lab member whom I would especially like to thank is Leo Volaskis for his involvement in the process, conducting the actual assays, and help in analyzing data. The members of Dr. Ghadiali's lab group were supportive in my research efforts and have given me valuable comments on my work.

Table of Contents

List of Figures

List of Tables

Chapter I. Introduction

Breast cancer affects hundreds of thousands of individuals each year, and it is the second most common form of cancer in women (1). In 2007 alone, approximately 202,964 men and women were diagnosed with breast cancer in the United States and 40,598 women died to the disease (breast cancer is rare in men and approximately 500 males die from breast cancer each year) (1). However, survival rates for different disease stages of breast cancer types differ greatly. Statistics gathered between 2001-2007 show that if the tumor is localized or regional to the lymph nodes at the time of diagnosis, the five-year survival rate is between 98.4-83.9% respectively (2). However, if the tumor has metastasized, the five-year survival rate drops to 23.8% (2). The difference in these odds elicits more investigation into the metastatic stage of breast cancer.

While much research has been conducted surrounding the biological aspect of cancer cells, little research quantifies how cancerous cells move, otherwise known as cell migration. Cell migration is the study of individualized cell movement through extension-contraction cycles. Migration is found in several physiological processes including embryonic development, immunity, wound healing, and cancer metastasis (3, 4). This process is summarized in Figure 1 and involves actin-myosin forces exerted by the cell and its interaction with the substrate (5). There are also several proteins involved, some of which include Cdc42 and Rac and Rho GTPase proteins. The directionality of the cell in the protrusion phase is influenced by Cdc42 by its coordination of actin polymerization with microtubule attachment at the leading edge (6). Rac is seen at the protrusion and adhesion phase and also promotes actin polymerization and new focal adhesions (6). Finally, Rho also plays a role in directing actin reorganization, but especially acts

to increase the actin contractility of the trailing cell body. The effect of proteins, substrate, and overall cell state controls the motility of cell migration.

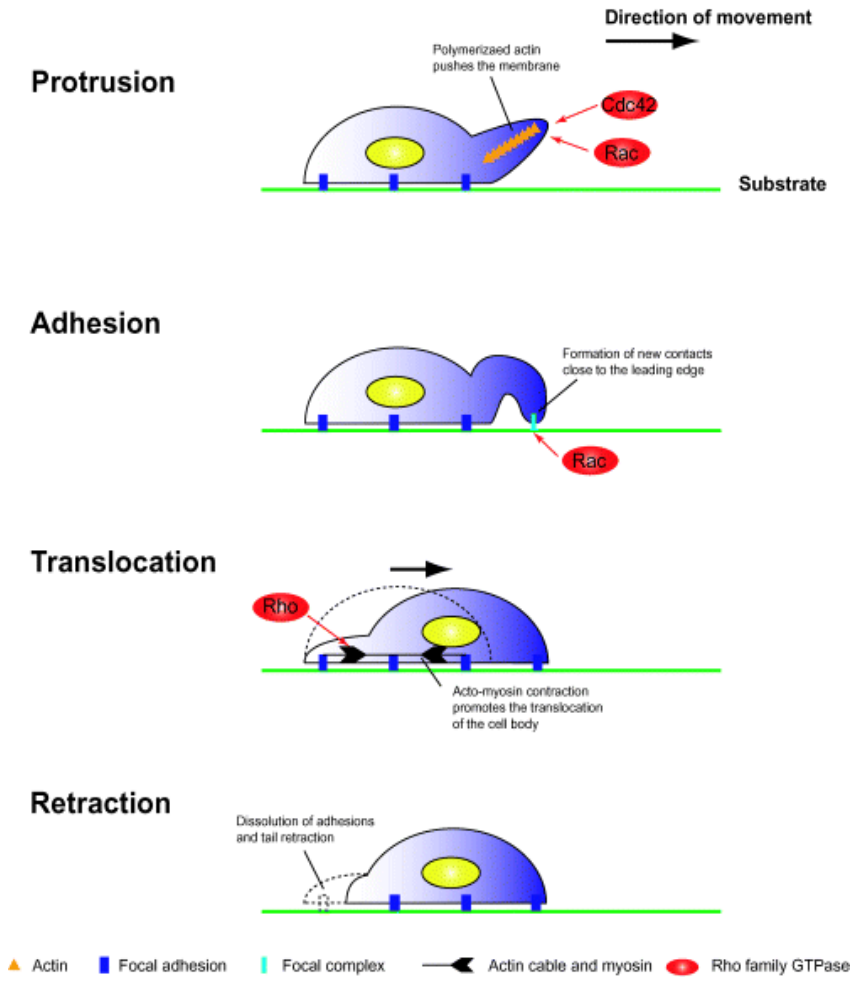

Figure 1: Schematic of Cell Migration Process

(3)

Several studies have shown over-expression of certain proteins in terms of cancer, including the aforementioned and myoferlin, a protein part of the ferlin family thought to be involved in breast cancer (3,7,8). Myoferlin expression influences the plasma membrane through membrane repair and receptor stabilization (7,8,9). Perhaps by altering the membrane structure, the cell can stretch through extension generation and thereby increase its surface area for locomotion (9). This implies that myoferlin might be involved in cell motility processes and invasion. Over-expression of myoferlin as well as other key proteins may play a direct role in cell motility, leading to increased cell migration.

This study investigates the motility patterns characteristic of invasive cells compared to noninvasive cells. When compared to noninvasive cells, there are several key attributes thought to be distinct to invasive cells. An epithelial to mesenchymal transition (EMT) is a proposed indicator of tumor progression that suggests a phenotype change of particular importance. In EMT, suppression of proteins responsible for cell adhesion like E-cadherin, results in a loss of cell-cell junctions (10). Cells that would normally move as a connected unit to close a wound, for example, detach from other cells and move as a single cell unit (3, 4, 10). This process is summarized in Figure 2, which also shows the typical epithelial to mesenchymal morphology change.

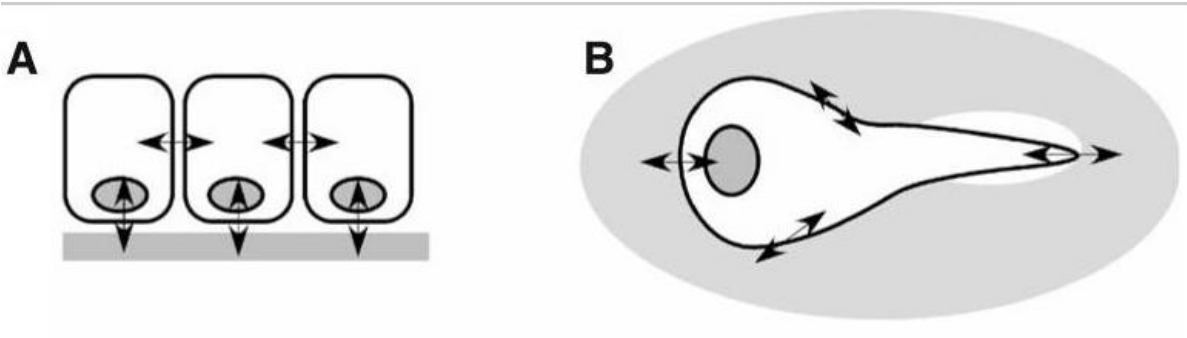

Figure 2: Comparing epithelial and mesenchymal morphology

A. Epithelial (cuboidal) morphology; arrows indicate mechanical force sensed from neighbor cells and substrate B. Mesenchymal (ameboid) morphology; arrows indicate force exchanged with substrate, protrusions evident in direction of motility (10).

The reverse process transitioning from a mesenchymal to epithelial cell, MET, is also possible (6). These experiments involve the protein myoferlin and removing over-expression as seen in the controls. It is hypothesized that cells with reduced, or more normal protein expression from a proliferative, invasive cell will behave with more epithelial characteristics.

Lastly, inhibiting cell motility at an early stage to increase the chances of survival has several therapeutic opportunities. With increased screening programs, the disease can be caught at an early, non-invasive stage. Even at an early invasive stage, treatments that reduce cell motility will decrease tumor margins involved (8, 10).

Chapter 2. Materials and Methods

This project analyzes human breast adenocarcinoma cells (MDA-MB-231). This line has an invasive phenotype and is known to proliferate on substrates known to indicate tumorgenecity like matrigel (12).

This study uses experiments comparing a negative (WT, n=3) and positive (LTVCTRL, n=3) control to myoferlin knockdowns (MYOFKD, n=3) MDA-MB-231 for a total of nine assays (13). A lentivirus infection was used to silence gene expression of the MYOFKD, resulting in decreased myoferlin expression (7, 9). Increased myoferlin expression has been shown to increase cell motility and migration patterns related to metastatic cancer (9). Each experiment was previously conducted and used for this study's analysis.

Cells were seeded in a monolayer and placed in a wound healing assay, which is used to study cell migration and cell interactions. A clean scratch is wiped down a confluent monolayer simulating a wound, and migration from both sides is documented for 24 hours. Images were captured every 10 minutes to track migration progress giving a total of 144 migration frames (13).

2. 1 Cell Tracking

Images showing the progress of migration were compiled into a video format to allow for superficial viewing of wound closure. Cells are tracked in video format using the Image J Manual Tracking plug-in available for download from NIH (11). The images in the video can be processed through additional downloadable plugins or existing imaging options to allow for better clarity. A minimum of 50 cells were tracked on either side of the wound. The number of cells tracked varies based on initial cell density on either side and due to time constraints. Cells with a completed track were numbered using the "overlay dots" option to separate cells into the

leading edge and those positioned deep towards each side and to make sure one cell was not tracked more than once. Table 1 reports the number of cells tracked for each experiment.

Table 1: Number of cells tracked for each experiment

(Includes cells on either side of the wound)

	MDA-MB-231								
Type	*WT*			*LTVCTRL*			*MYOFKD*		
Date of experiment	1/28/2011	2/17/2011	3/28/2011	2/27/2011	4/2/2011	4/4/2011	2/18/2011	2/22/2011	4/3/2011
Number of Cells tracked	100	130	100	130	100	100	140	130	130

Cells were tracked with two goals in mind. To get an accurate representation of both the leading and trailing edge, an average of 20 cells were tracked in the leading edge while the rest in the sample were tracked in the trailing portion. A cell was classified in the leading edge if it was approximately three cells or less deep into the leading edge at the beginning of each experiment. Another goal included the importance of cell-cell interactions during migration. To capture the potential of this cell-cell interaction, each cell tracked has at least two tracked neighbor cells.

The cell tracking program records the coordinates of the cell, distance travelled between each 10 minute interval and the velocity every 10 minutes. After all cells from one experiment are tracked, the original data file from Image J automatically saves as “Results” with the original file name following. The left and right side cell tracks were saved individually for ease of use. These data tracking results for each side of a wound closure assay can be uploaded in a software analysis, Chemotaxis and Migration Tool 2.0 available from ibidi® (14).

Chapter 3. Chemotaxis Parameter Analysis

3.1 Definition of Parameters

After tracking an experiment, the results file must be initialized in the Chemotaxis analysis software to account for 10μm/9 pixels and the time interval between each image, 10 min. Four parameters were taken into account to capture any differences between the knockdown and controls cells. They include: velocity, distance (accumulated and Euclidian), directionality, and x-forward migration index (FMI_x). Velocity, directionality and FMI_x are defined in Equations 1-4.

$$Velocity_i = \frac{d_{i,accum}}{t_{i,total}}$$

$$Equation\ 1: for\ 1 \leq i \leq n$$

$$where\ i\ is\ the\ in\ index\ of\ a\ single\ cell\ and\ n\ \ is\ the\ total\ number\ of\ cells\ tracked$$

$$and\ d_{i,accum} is\ the\ accumulated\ distance$$

$$Directionality_i = D_i = \frac{d_{i,euclid}}{d_{i,accum}}$$

$$where\ d_{i,euclid}\ \ is\ the\ straight\ distance\ between\ the\ beginning\ and\ end\ coordinates.$$

$$D_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{d_{i,euclid}}{d_{i,accum}}$$

$$Equation\ 3: Average\ directionality\ for\ all\ cells\ tracked\ in\ one\ experiment.$$

Directionality is scaled from 0 to 1 (Figure 3) where mesenchymal cells are hypothesized to have a lower directionality. Single-cell migration patterns assume cells break their junctions via the loss of cadherins and increased expression of MMPS, which weaken cell-ECM attachments (17,18). It is thought that single cells will have more freedom and respond to wherever the gradient or attractants move throughout the culture.

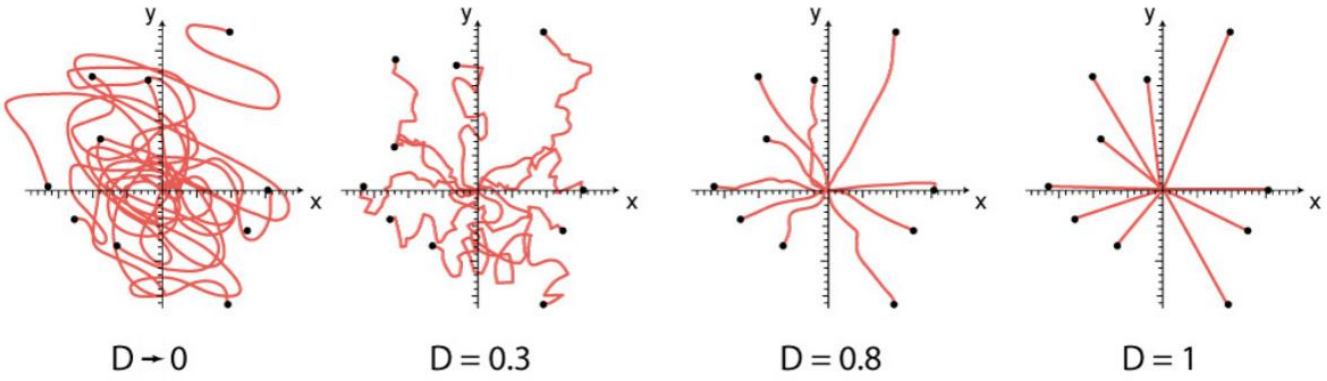

Figure 3: Scale of directionality

Mesenchymal cells are thought to exhibit a lower directionality compared to cells with an epithelial morphology (13,14)

Forward migration index in the x direction (FMI_x) is similar in concept to the directionality parameter, but quantifies movement in only the x direction. This is the direction in which cells migrate to close the wound. The same differences mentioned for directionality between mesenchymal and epithelial cells are expected for FMI_x. This parameter is defined by equation 4.

$$FMI_x = \frac{1}{n}\sum_{i=1}^{n}\frac{x_{i,end}}{d_{i,accum}}$$

Equation 4: Average FMI_x for all tracked cells. $x_{i,end}$ is the total distance in the $x-$direction only for one cell

3.2 Chemotaxis Parameter Results

The parameter results for the 231 cell line are seen for each experiment in Figures 4-8. The averages of all cell tracks in one experiment are reported here with standard error (SEM) bars and significance from the results of a two sided t-test.

3.2.1 Velocity

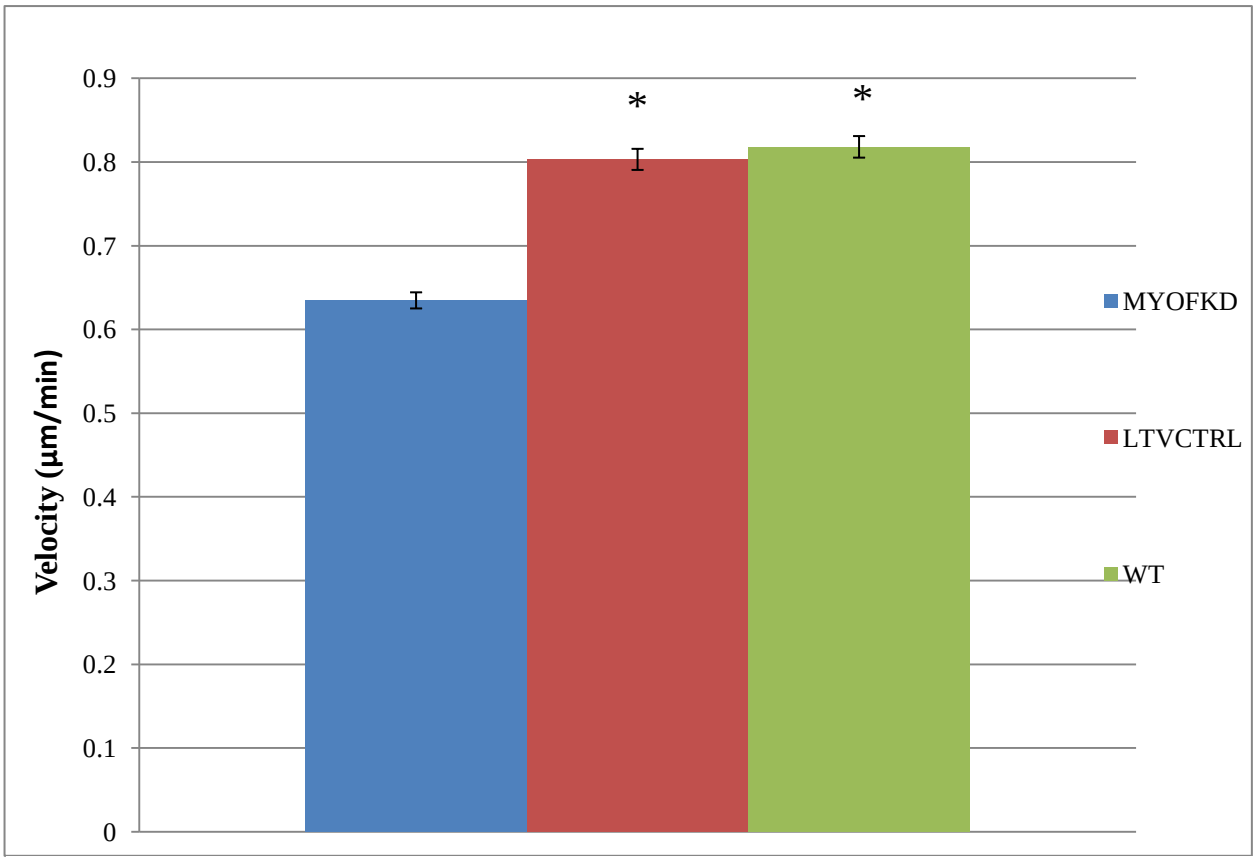

Figure 4: Average velocity
(all n=3); SEM shown (MYOFKD=0.00963, LTVCTRL=0.01261, WT=0.01289; 2-sided equal variance t-test (* $p<0.0001$ between each control and MYO-KD)

The velocity of the knockdown cells is significantly less than that of both controls. The average of lentiviral and wild type controls are very similar. These results support the hypothesis that cancer cells are more aggressive, and thus move and proliferate more quickly (6, 10,16). A number of proteins that are over-expressed that facilitate a morphology change could allow for cells to move more quickly. An increase in MMPs as seen in several cancer types could be responsible for creating a path of less resistance for cells to migrate (16,17)

3.2.2 Accumulated and Euclidian Distance

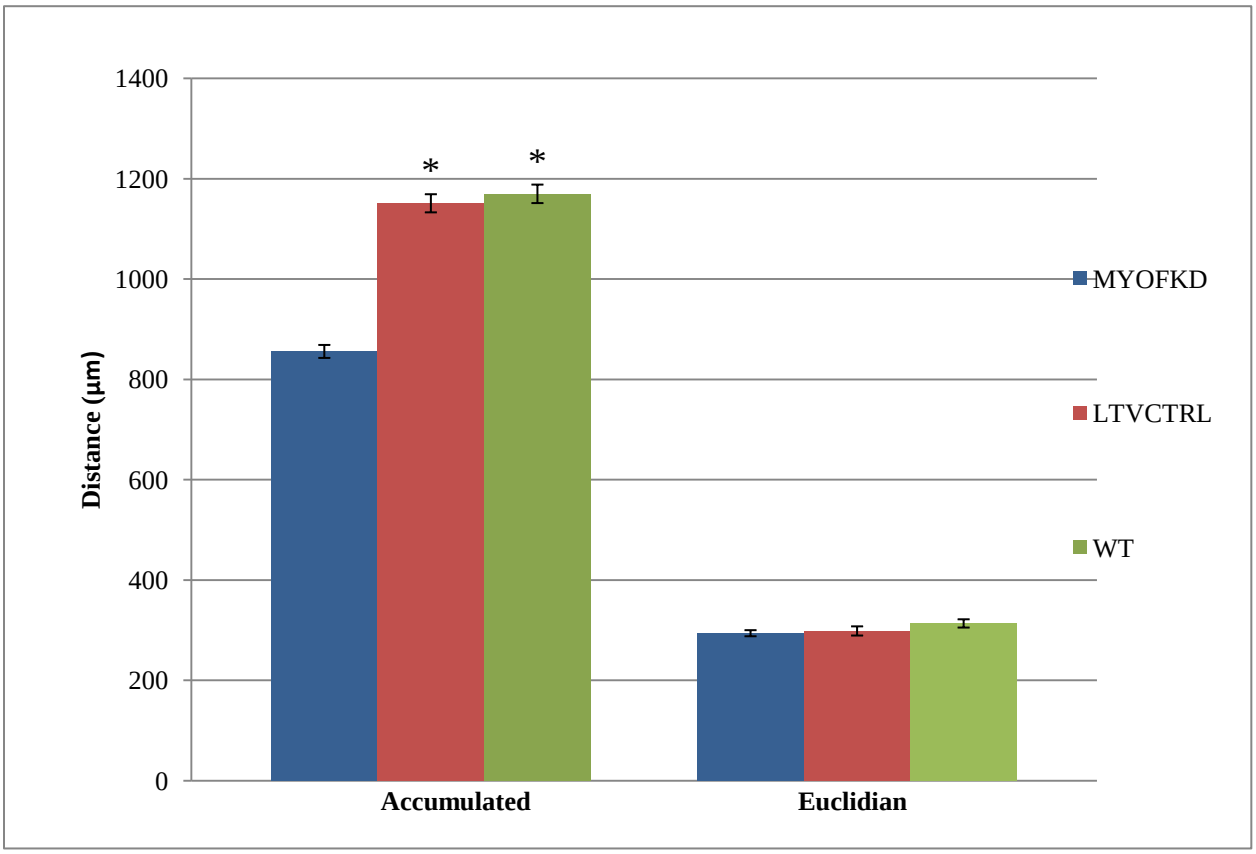

Figure 5: Average accumulated and Euclidian distance
(all n=3); SEM shown as standard error bars; 2-sided equal variance t-test (* $p<0.05$ between each control and MYO-KD)

The knockdown cells collect less accumulated distance compared to the controls. This increase in distance travelled supports the notion that mesenchymal cells can travel farther and acquire more ability to invade surrounding tissue (15). As seen in the results for velocity, the lentiviral and wild type cells experience a similar overall average whose difference is not statistically significant. It is interesting to note that all three cell types have similar Euclidian distances. This similarity will affect changes in directionality between all three cell types.

3.2.3 Directionality and FMI_x

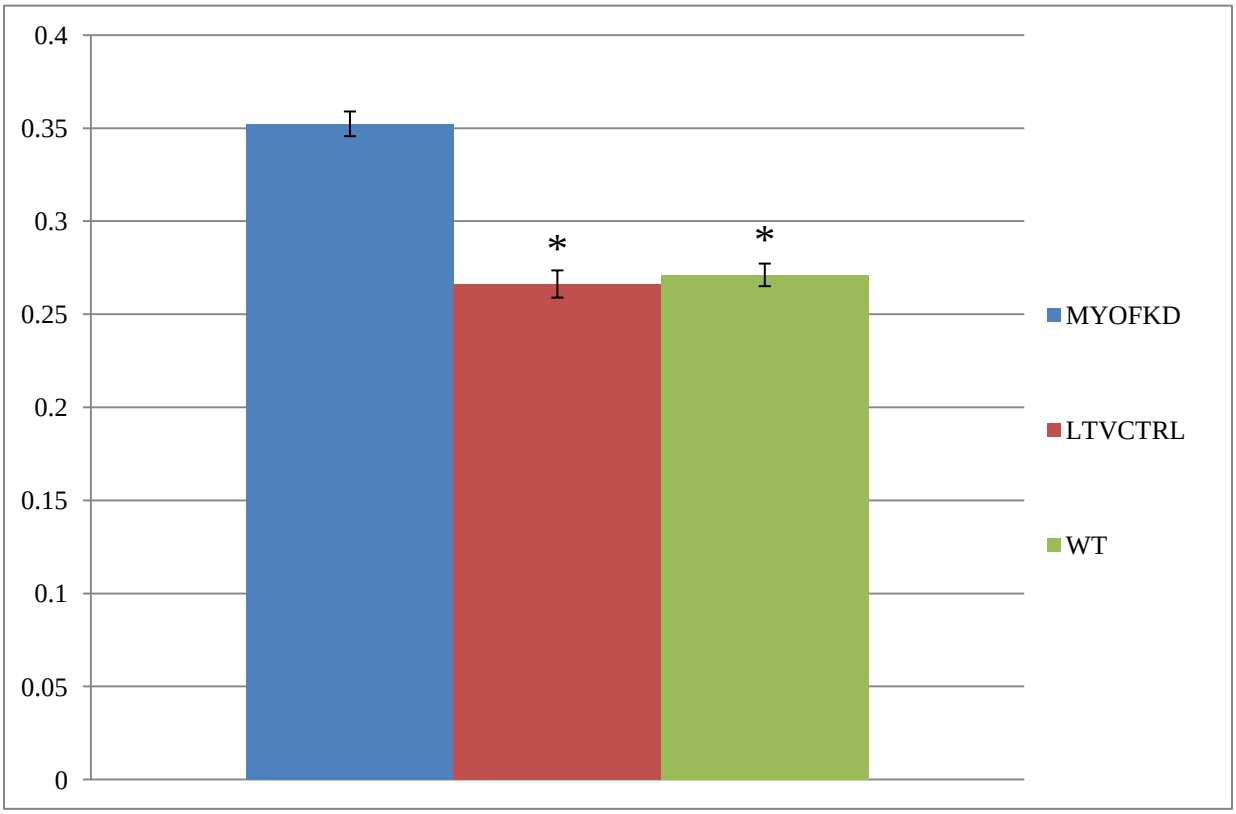

Figure 6: Average directionality
(all n=3); SEM shown as standard error bars; 2-sided equal variance t-test (* $p<0.005$ between each control and MYO-KD)

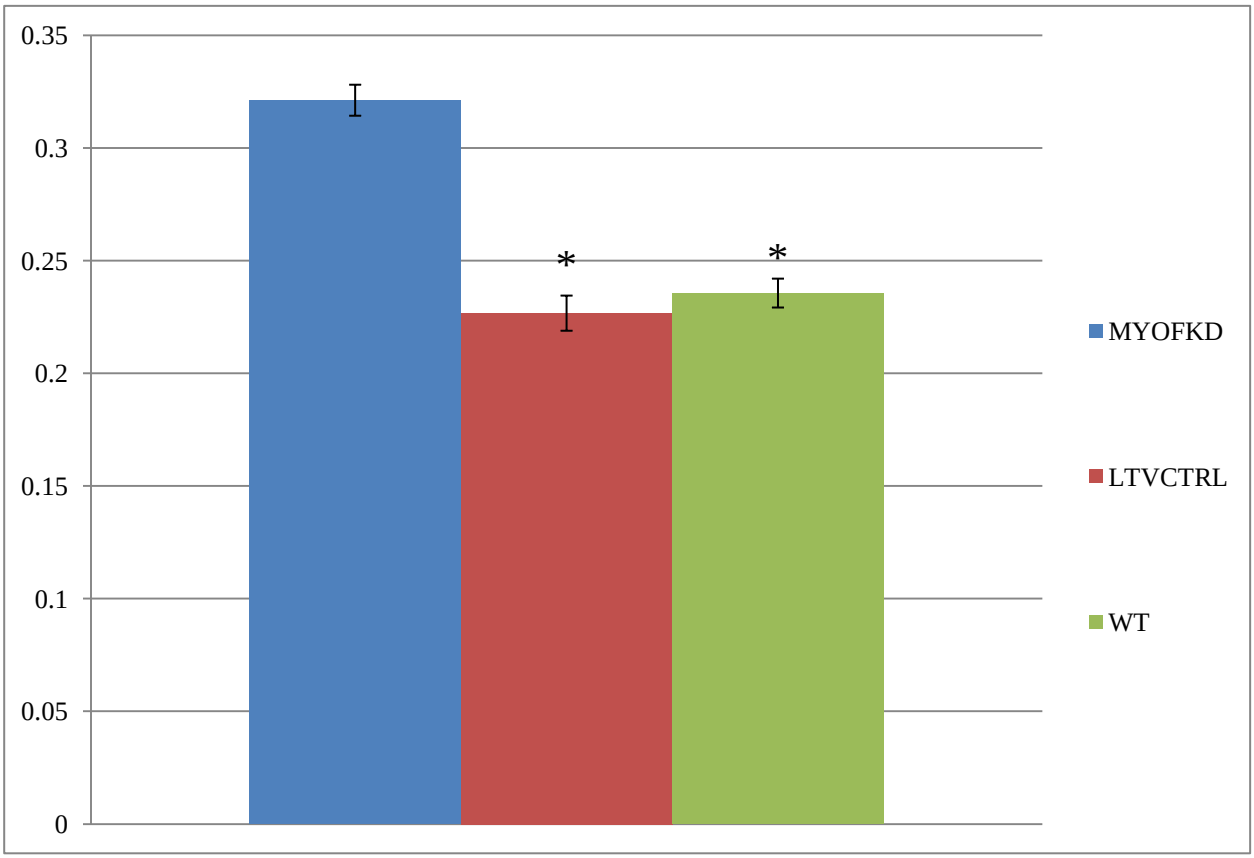

Figure 7: Average $|FMI_x|$
(all n=3); absolute values for the right side of each experiment are used for averaging since cells on the right move in the negative x direction. SEM shown as standard error bars; 2-sided equal variance t-test (* $p<0.005$ between each control and MYO-KD)

There is a difference in both directionality and FMI_x between the controls and the knockdown cells. The knockdown cells experience a higher directionality and FMI_x whereas both controls experience similar, but decreased directionality and FMI_x. These results support the notion that more directional cells exhibit an epithelial phenotype and cells with cancerous characteristics share a mesenchymal phenotype (15). This difference can be clearly seen in a plot comparing left side cell tracks of one knockdown experiment to that of the lentiviral control and wild type cells (Figure 8).

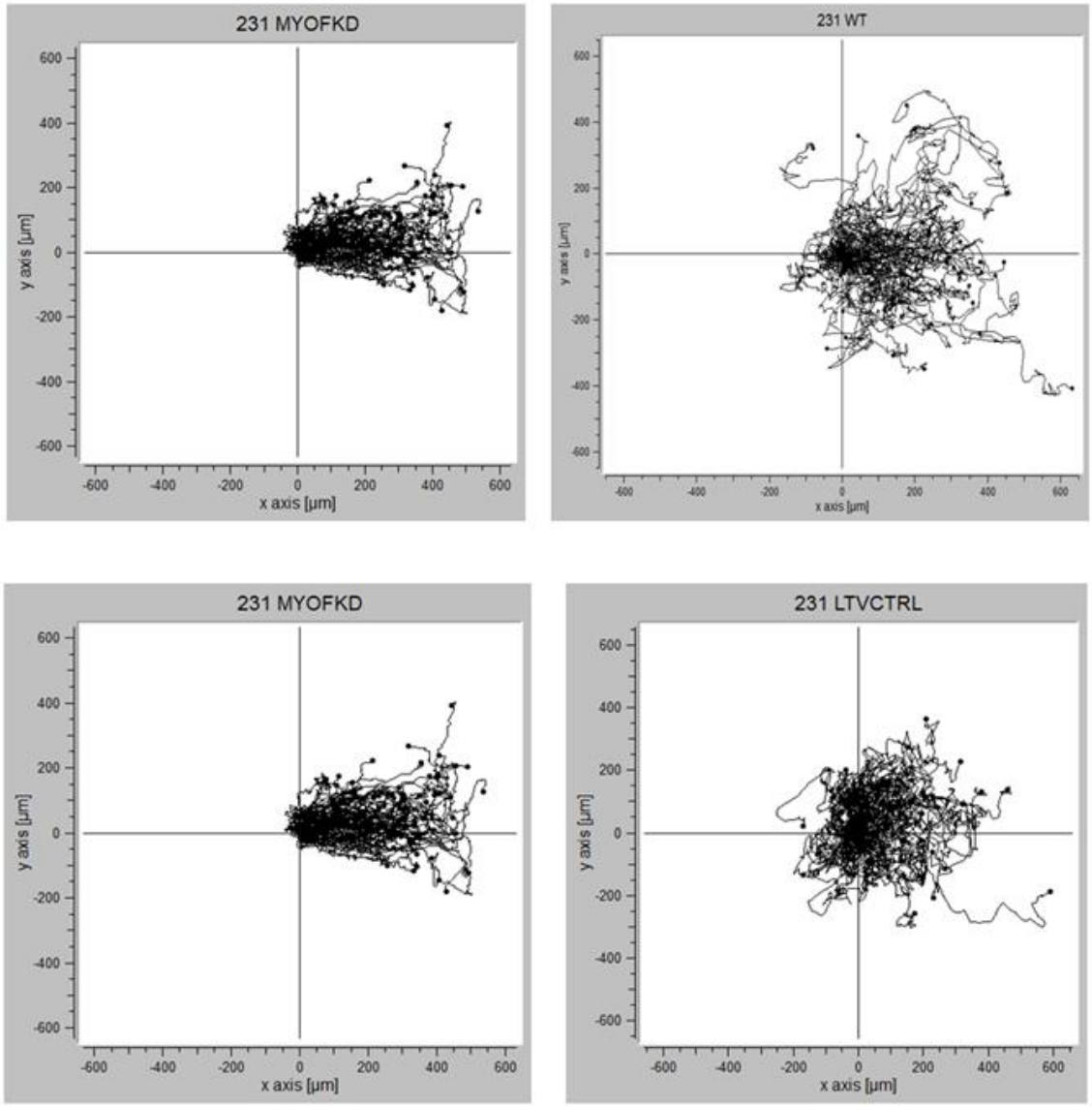

Figure 8: Cell tracks comparing difference in directionality
Myoferlin left side accumulated cell tracks (n=50) exhibit more directionality compared to the left side of the wild type (n=50) and lentiviral controls (n=50)

Chapter 4. Introduction to Correlation Models

Results from the 231 experiments reflect the expected differences between the epithelial phenotype in the knockdown cells and the mesenchymal phenotype in the wild type and lentiviral controls. However, the results comparing the averages between the experiments of each parameter do not fully capture which cells detach from an adjacent cell, nor how many and how cells migrate. This prompted the need for a model which attempts to fill that void: to estimate and predict if cells detach from their neighboring cells during EMT. This model is also used to reinforce the differences and significance seen in the parameters quantified above. These models focus on the interactions between the target and neighbor cell using a combination of direction, distance, and time. The code for this model can be seen in Appendix A.

4.1 Correlating Direction Using Euclidian Distance

4.1.1 Methods and Results

To develop a model, the original results file from tracking was saved as a text file in MATLAB. For consistency, each original results data file was named with following standard: celltype_date_file1_left (or right) with each side of an experiment loaded separately.This file is the variable "data" in MATLAB code. In order to determine if a cell detaches from other cells, a cell's neighbor's must first be identified. "File 2", or the variable "cellneighbor" in MATLAB is the cell-neighbor cluster matrix the user should manually input after each experiment is tracked. A cell-neighbor cluster takes a cell as a target and lists its neighbors in an array, those that initially touch the target cell, in file 2. Each cell tracked is considered a target cell, and the file must be constructed in increasing number of neighbors. Subsequent zeros follow the last

neighbor listed to allow MATLAB to read a correctly dimensioned matrix, and they also represent that no other neighbors exist for that target cell. Each side of the wound is called separately, but the program runs values for each side simultaneously.

This model compares the direction between a cell and its neighbors using each cell's Euclidian distance. In this sense, this model attempts to discriminate between collective cell and single cell migration. Collective cell migration in this scenario assumes the Euclidian direction for a cell and its neighbor retain cell-cell adhesions and migrate to similar locations. In contrast, single cell migration will see large differences in direction due to detachment from the wound edge, especially cells from the leading edge.

This model was developed using the simple notation of the dot product. Correlation values, which quantify if cells are moving collectively or as a single cell, are calculated using normalized unit vectors of the Euclidian distance. These values range from 0-2, with 0 implying movement in the same direction, 1 implying the target and neighbor have perpendicular Euclidian directions and 2 meaning movement in the exact opposite direction

$$C = 1 - dot\big((t_x, t_y), (n_{i,x}, n_{i,y})\big)$$

$Equation\ 5: Correlation\ parameter,C, comparing\ the\ direction\ a\ neighbor\ to\ a\ target\ cell.$

$$t_x, t_y\ are\ the\ normalized, euclidain\ end\ coordinates\ for\ target\ cell$$

$$n_{i,x}, n_{i,y}\ normalized, euclidan\ end\ coordinates\ for\ neighbor\ cell\ i\ where\ 1 \leq i \leq n_{neigh,total}\ (total\ \#\ neighbors)$$

The code calls the appropriate coordinates for each neighbor and the target cell being analyzed and computes the average value of *C* for the total number of neighbor cells for one target cell (Equation 6).

$$C_{cluster} = \frac{1}{n_{total}} \sum_{i=1}^{n_{neigh,total}} 1 - dot\big((t_x, t_y), (n_{i,x}, n_{i,y})\big)$$

Equation 6: Average correlation value between a target cell and each of it'sneighboring cells.

Each cell tracked is considered a target cell and its neighbors are specified in the cell-neighbor matrix, File 2 described previously in methods of developing the model. The average for each $C_{cluster}$ value for each target cell and its neighbors are calculated for the left and right side of the wound separately. The left and right side $C_{cluster}$ values are combined into a single matrix and averaged to find an overall correlation value comparing the direction travelled and final position of a cell in relation to its neighbor for all three experiments (Equation 7). The average values (n=3) for the knockdown and controls are reported in Table 2 and Figure 9.

$$C_{left\ and\ right} = \frac{1}{n_{total}} \sum_{i=1}^{n_{total}} \frac{\sum_{i=1}^{n_{neigh,total}} 1 - dot\big((t_x, t_y), (n_{i,x}, n_{i,y})\big)}{n_{neigh,total}}$$

Equation 7: Average correlation value for one side of an experiment for $1 \leq i \leq n_{total}$ $(total\ number\ of\ clusters, left\ and\ right)$

Table 2: Correlation values of the direction using Euclidian distance model

Direction Correlation Model			
Cell type	MYOFKD	LTVCTRL	WT
Correlation, *C*	0.1681	0.3690	0.2744
Standard deviation	0.2579	0.4601	0.3245
SEM	0.01342	0.02610	0.01837

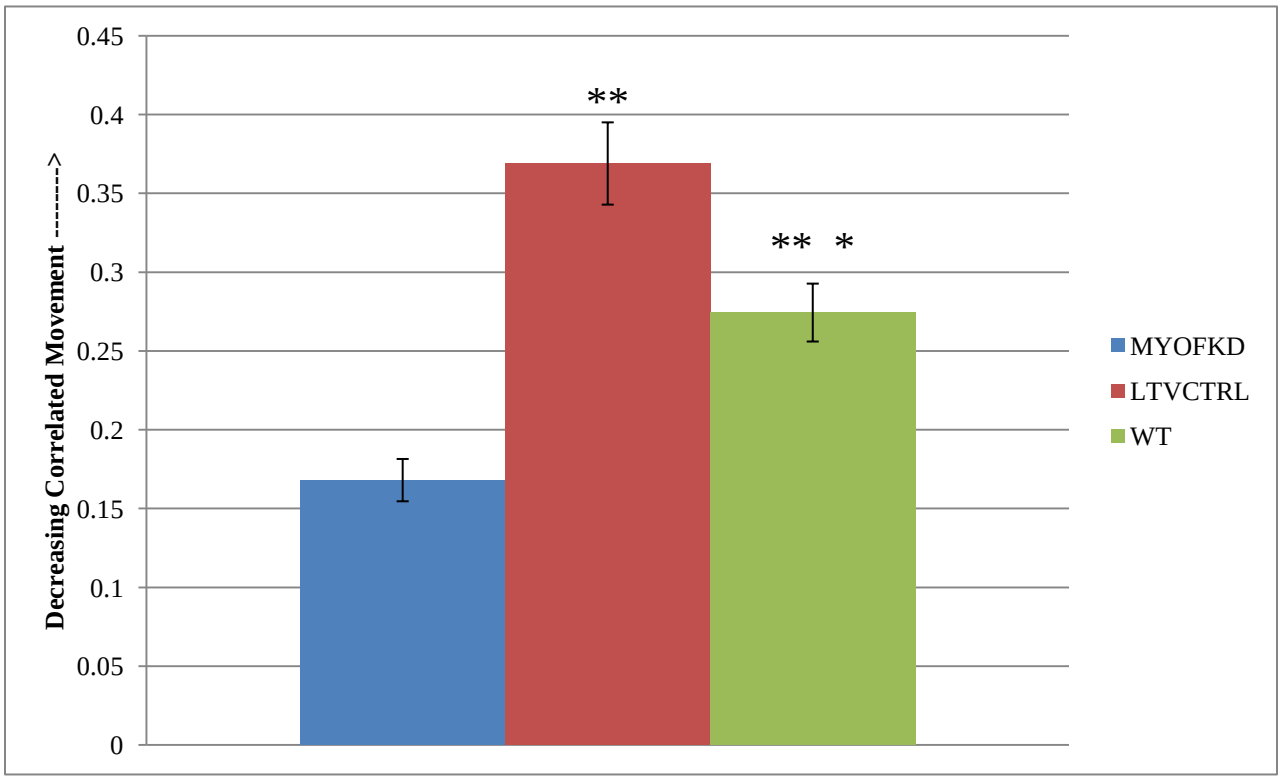

Figure 9: Euclidian Distance Correlation graph
Standard error bars shown; **p<0.0001 2-sided equal variance t-test between controls and MYOF-KD; *p<0.05 between both controls

4.1.2 Discussion

The results from the correlation model support the notion of the myoferlin knockdown cells being more correlated with respect to direction. Both controls show larger correlation averages that are significantly different. Thus, this model captures differences in more single cell migration as noted by the wild type, especially since both controls have similar directionality. For both controls, it is apparent that the neighbor cell compared to the target cell travels in a more different direction. The large standard deviation for all three cell types show great variety in directions travelled, which is supported from the distance parameter graph (Figure 5). Both controls have a higher correlation value and have a lower directionality, meaning they have greater difference in direction travelled and they travel in a more random fashion. This could help to explain the phenomenon of cells breaking from the sheet trying to close the wound. Overall, a greater difference in correlation averages between each cell type would instill more confidence in the model in predicting a more epithelial or mesenchymal phenotype. A second model described below attempts to create a greater distinction and further verify the results of this model.

4.2 Correlating Direction Using Euclidian Distance with Time Step

4.2.1 Methods and Results

The second model attempts to quantify time and break down the accumulated pathway of each cell into five time frames. The same calculation (Equation 6) is used as seen in the previous model, but is calculated in five separate time frames for one cell track. Averages for the left and right side for each cluster in each time frame are averaged for all three experiments. Specific values are shown in Table 3 and graphical comparisons in Figure 11. This approach captures

more of the accumulated pathway taken by the cell especially if the cell has low directionality. The difference between the previous model and this model taking into account time can be seen in Figure 10. The full code for this model can be seen in Appendix B.

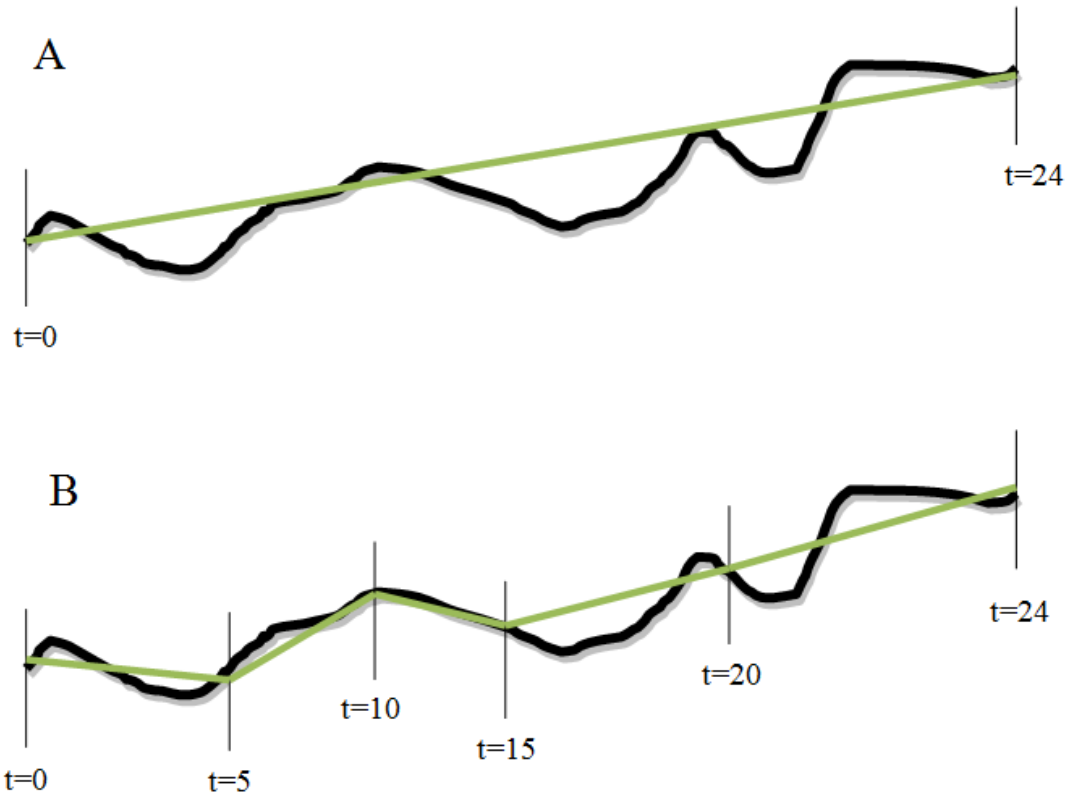

Figure 10: Example cell path showing the effect of time steps when correlating direction travelled.
Compares the difference between time distance model (B) compared to the model using the Euclidian distance for the entire time length (A). Curved line is the actual cell path, straight line is the Euclidian distance between time frames.

Table 3: Correlation values for model using Euclidian distances and time steps

Directional Model with Time-Steps			
Cell type	MYOFKD	LTVCTRL	WT
Correlation, *C*	0.4648	0.5950	0.5923
Standard deviation	0.4617	0.5149	0.4977
SEM	0.01075	0.01304	0.01236

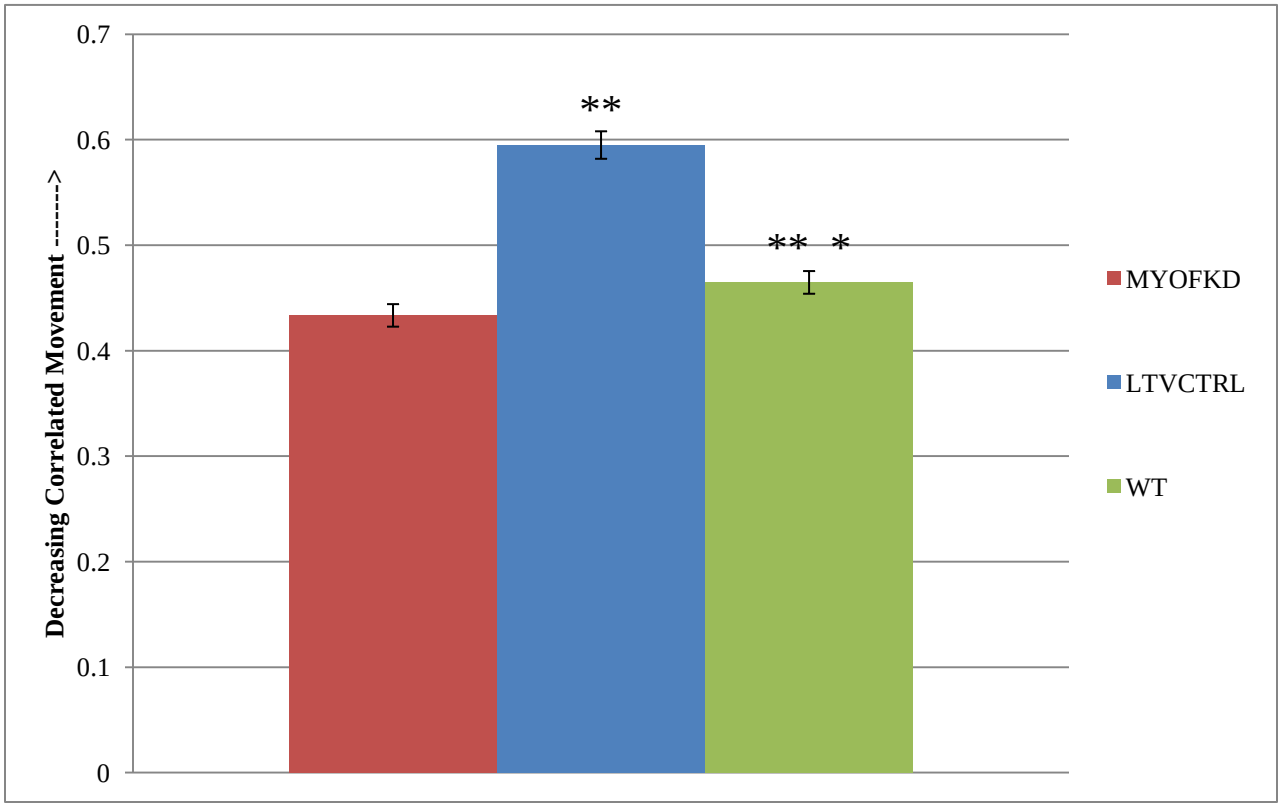

Figure 11: Correlation for Time Step Model
Standard error bars shown; **p<0.0001 (between LTV-MYOF-KD and LTV-WT) *p<0.05 (between MYOF-KD –WT) 2-sided equal variance t-test.

4.2.2 Discussion

This model captures more of the accumulated pathway of each tracked cell. Therefore, it was hypothesized that this model would show a greater difference between the knockdown and the control cells. However, the greatest difference in the previous model is approximately 0.2009 whereas the difference between this model is only 0.1302. This model suggests there is less difference in migration pattern than previously expected between an epithelial and mesenchymal phenotype.

This model does support the idea of consistency in predicting overall differences between the controls and knockdowns. The myoferlin experiments have consistently lower values, meaning they have less difference in direction between the neighbor and target cell. The lentiviral and wild type cells are consistently higher in correlation values, meaning they are less correlated by having more difference in directional movement. There is a significant difference between both controls as seen in the previous model. There is consistency showing that while the WT move with similar directionality, there is more difference in direction travelled between the target and neighbor cell; thus, this still shows more single cell migration patterns. More standard deviation is reported, implying more variety in correlated or uncorrelated directional movement between the target and neighbor cell.

So far, the two models introduced using the Euclidian distance to compare the direction travelled have only compared the direction between a target and each of its neighbors. To fully consider correlated movement, distance along with direction ought to be modeled. A neighboring cell may travel in the same direction as its neighbor cell, but may not travel approximately the same distance as the target cell. Distance and direction are incorporated into one model that is described below.

4.3 Correlating Direction and Distance Using Time Steps

4.3.1 Methods and Results

The third and last model attempts to account for the difference in distance and direction. This model is based off the law of cosines and finds the gap distance between the end points of the target and neighbor cell for every five time frames (as seen in the previous model). This is represented in Figure 12.

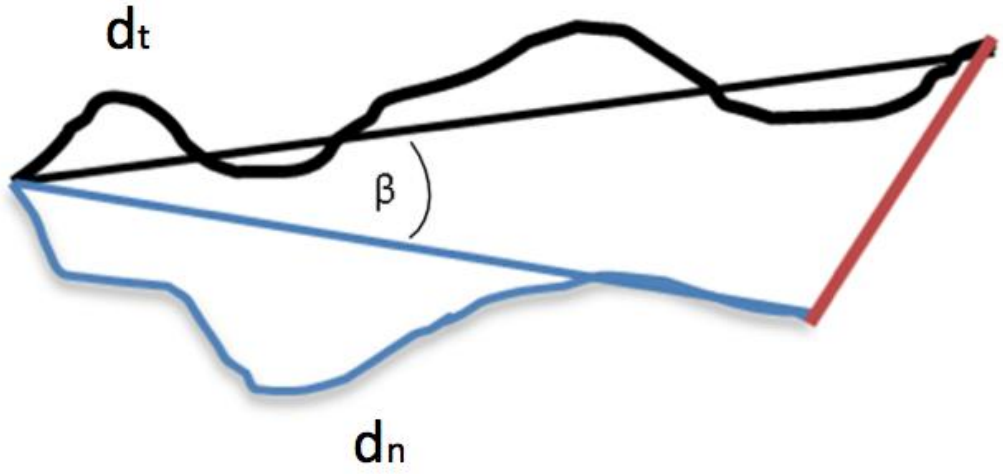

Figure 12: Example cell path showing model that uses direction and distance. Normalized to the origin taken from 0-5 hrs. Black curved line (d_t) is the target cell path, blue curved line (d_n) is neighbor cell path. Straight lines are the Euclidian distances, respectively. The red line is the gap distance calculated.

By definition, the law of cosines tells the gap distance to be equal to:

$$d_g^2 = d_t^2 + d_n^2 - 2(d_t d_n \cos\beta)$$

Equation 8: d_g *is the gap distance,* d_t *is the euclidian distance of the target cell,* d_n *is the euclididan distance of the neighbor cell and* β *is the angle between them*

$$\cos\beta = \frac{dot((t_x, t_y), (n_x, n_y))}{|d_t| * |d_n|}$$

Equation 9: (t_x, t_y) *are the target end coordinates for each time frame,* (n_x, n_y) *are the neighbor end coordinates for each time frame.*

Substituting the definition of the dot product (Equation 9) into the law of cosines and dividing by the length of the target cells gives the final gap distance correlation equation (Equation 10).

$$C_{gap\ distance} = \frac{\sqrt{d_t^2 + d_n^2 - 2(dot((t_x, t_y), (n_x, n_y))}}{|d_t|}$$

Equation 10: *Final correlation gap distance equation. Gap distance is divided by the length target cell to normalize the gap distance to a specific length*

Results of the correlation gap distance model can be any positive number, but the closer the value is to 0, the more correlated the movement of neighbor to target in direction and distance. These results are calculated using the concept of the time step model above breaking cell paths into five time frames. An important distinction to note is not only the difference in equations between the previous and currently discussed model, but also that the time step model uses unit vectors from the Euclidian distance of a cell and the current model does not use unit

vectors. Correlation values are averaged over every time frame cluster average for both the left and right side of all three experiments. Values are reported in Table 4 and shown in Figure 13.

Table 4: Correlation values for the direction and distance model.
Standard deviation and errors were reported taking into account the number of clusters averaged for both the left and right side of all three experiments.

Direction and Distance Model			
Cell type	MYOFKD	LTVCTRL	WT
Correlation, *C*	1.3778	1.4662	1.5161
Standard deviation	2.3703	1.5837	2.0671
SEM	0.05518	0.04009	0.05234

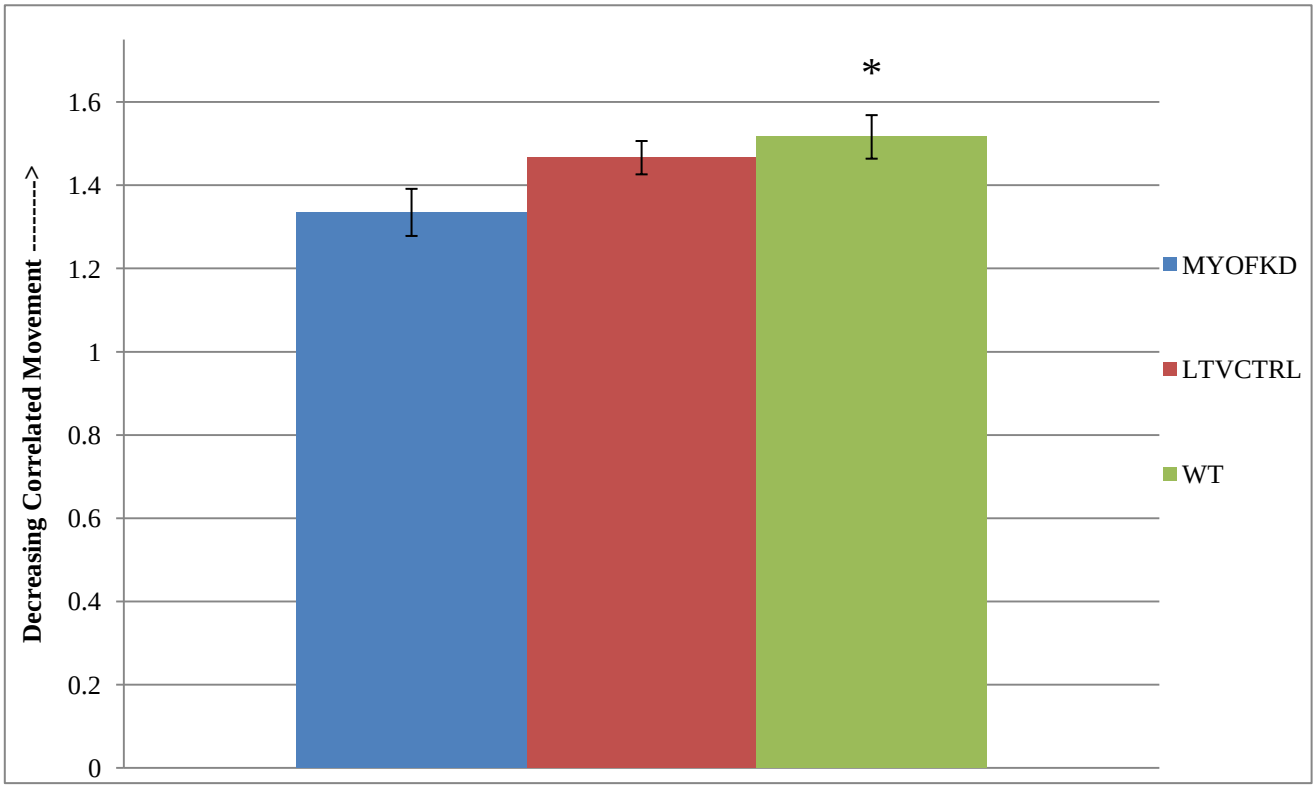

Figure 13: Correlation Averages for Model Using Direction and Distance Time Steps
Standard error bars shown; *p<0.05 (between MYOF-KD and WT) 2 sided equal variance t-test

4.3.2 Discussion

The results for the knockdown cells imply that the gap distance is on average, 38% larger than the Euclidian distance of the target cell. The average gap of the lentiviral cells is about 47% longer than the Euclidian distance of the target cell and the wild type has the largest gap distance, with the distance being about 53% larger than the Euclidian distance of the target cell. Thus, the knockdown cells are more correlated than the controls. A limitation of this model lies in the results possible. The neighbor cell is considered more correlated to the target cell as the correlation value approaches zero. The other extreme average value for uncorrelated movement is not defined; rather, the results tie back to how little of a gap exists between target and neighbor cell in each time frame.

This model further verifies the consistency seen in the previous models in predicting the correlation seen in the knockdown cells. Though the difference in correlation average between the controls is small, it is interesting to note that the wild types are the least directional. The previous models have shown the lentiviral control cells being the least directional. It is implied that the wild type have a very similar average difference in direction compared to the lentiviral controls, but do not travel as far as their target cell. The large standard deviation is consistent across this model as in the previous two. This is also supported in the standard deviations from the accumulated and Euclidian distance graph.

Chapter 5. Conclusions and Recommendations

Cell migration is a key process involved in many biological events, like embryonic development, wound healing, immunity, and tumor metastasis. An epithelial to mesenchymal transition is a proposed method of tumor progression that argues for an epithelial to mesenchymal phenotype change. In this way, cells alter their migration patterns to become more invasive. The same argument can be made for a mesenchymal to epithelial transition, with cells changing phenotype to become less invasive. However little research quantifies these processes and the mechanism is poorly understood.

The results comparing velocity, direction, directionality, and FMI_x are consistent with the notion that the knockdown cells display a more epithelial morphology and phenotype whereas the controls are consistent with an invasive cancer cell. Results from the models show consistency when comparing the knockdown cells to both control cells. On average, the knockdown cells have more correlated movement when comparing the direction or distance between the neighbor and its target. These models show some predictive power when comparing correlated, sheet like movement to random, single cell migration.

We had hoped to find a more distinct difference with the directional model taking time steps and the direction and distance model with time steps. A limitation of the direction and distance model is the lack of a set range of values. While more correlated movement is associated with that of a value close to 0, the value at which a distinction between uncorrelated movement and correlated movement is somewhat unclear. To correct this uncertainty, a modification could be made to divide the gap distance by the target cell or vice versa, depending on whichever distance is larger. This would give a more clear ratio from 0-1, where 0 still represents more correlated movement, telling the user how great of a difference exists between a

cluster of cells. A more accurate representation of the end gap distance might not be best represented by normalizing the starting point of each cell to the origin. Rather, taking the ratio of the gap between the starting position of each target to neighbor cell to the gap distance between the end points of both cells could give a better representation of how of much difference in distance each pair accumulated.

A limitation of the first two models is that they do not take into account possible differences in distance travelled. Also, the first model doesn't capture the full accumulated pathway of the cell. The second model attempts to represent more of the accumulated pathway, but there are still some gaps in the pathway since it is only taken every five hours. To capture more accumulated pathway, more frequent time steps could be used. The time step models can be easily modified to allow the user to pick what time frames need to be analyzed and allow for more frequent time steps to be used. Thus, the time step models are user friendly and easily modified based on different user preference.

One last limitation includes the fact that each model doesn't account for the possibility that cells have formed new adhesions to another cell if it has attached. A modification used for future work in each model could incorporate a conditional statement that tells the program to stop correlating a cell to its neighbor if the gap distance or directional angle exceeds a certain value. This condition could be quantified and correlated back to single cell detachment. Additional future work also includes comparing the leading edge of each side of the wound to the body of each side. This could capture less correlated movement, extrapolating into single cell migration.

References

1. "CDC - Cancer - Statistics by Cancer Type." Centers for Disease Control and Prevention. National Center for Chronic Disease Prevention and Health Promotion, 10 Nov. 2010.

2. Howlader N, Noone AM, Krapcho M, Neyman N, Aminou R, Waldron W, Altekruse SF, Kosary CL, Ruhl J, Tatalovich Z, Cho H, Mariotto A, Eisner MP, Lewis DR, Chen HS, Feuer EJ, Cronin KA, Edwards BK (eds). SEER Cancer Statistics Review, 1975-2008, National Cancer Institute. Bethesda, MD.

3. Yamazaki, D, Kurisu, S and Takenawa, T Regulation of cancer cell motility through actin reorganization. Cancer Science 96, (2005): 379–386.

4. Kalluri, Raghu. "EMT: When Epithelial Cells Decide to Become Mesenchymal-like Cells."*Journal of Clinical Investigation* 6.119 (2009): 1417-419.

5. Friedl, Peter, Yael Hegerfeldt, and Miriam Tusch. Collective Cell Migration in Morphogenesis and Cancer. Int. J. Dev. Biol 48 (2004): 441-49.

6. Sahai, Erik. "Mechanisms of Cancer Cell Invasion. Current Opinion in Genetics & Development, 15.1 (2005): 87-96.

7. Eisenberg, Marisa C., Yangjin Kim, Ruth Li, William E. Ackerman, Douglas A. Kniss, and Avner Friedman. "Mechanistic Modeling of the Effects of Myoferlin on Tumor Cell Invasion." *National Center for Biotechnology Information*. U.S. National Library of Medicine, 3 Nov. 2011.

8. Li, Ruth. "The Expression and Effect of Myoferlin Depletion in Breast Cancer Cells." Diss. The Ohio State University, 2011. *OhioLINK ETD Center*.

9. Cipta, Stephanie, and Hemal H. Patel. "Molecular Bandages: Inside-out, Outside-in Repair of Cellular Membranes. Focus on “Myoferlin Is Critical for Endocytosis in Endothelial Cells”." *American Journal of Physiology: Cell Physiology* (2009).

10. Kumar, Sanjay, and Valerie M. Weaver. "Mechanics, Malignancy, and Metastasis: The Force Journey of a Tumor Cell." *Cancer Metastasis Review* 28 (2009): 113-27.

11. Rasband, Wayne. ImageJ: Image Processing and Analysis in Java. *National Institute of Health*.

12. “Product Data Sheet: MDA-MB-231/GFP Cell Line” Cell Biolabs, Inc. 2009.

13. Research conducted by Leo Volakis, The Ohio State University

14. Trapp, Gerhard, and Elias Horn. Chemotaxis and Migration Tool. *Ibidi: cells in focus*.

15. Yang, Jing, and Robert A. Weinberg. "Epithelial-Mesenchymal Transition: At the Crossroads of Development and Tumor Metastasis." *Developmental Cell* 14.6 (2008): 818-29.

16. Kalluri, Raghu, and Robert A. Weinberg. "The Basics of Epithelial-mesenchymal Transition."*Journal of Clinical Investigation* 119.6 (2009): 1420-428. 1 June 2001

17. Suresh, Subra. "Biomechanics and Biophysics of Cancer Cells." *Acta Materialia* 55.12 (2007): 3989-4014. Elsevier Ltd.

18. Fotey, Ramsey A., and Malcolm Steinberg. "Cadherin-mediated Cell-cell Adhesion and Tissue Segregation in Relation to Malignancy." *Int Journal of Dev Bio* 48 (2004): 397-409

Appendix A: Directional Model Using Euclidian Distances

```
      Calculates both sides of one experiment at one time

%% LEFT SIDE
%Correlating Cell-Neighbor Movement Direction Using Euclidian Distances
%Created by J. Sanders on 8/19/2011

%% Loading Data
data=load('WT_3_28_file1_left.txt'); %original file from Image J manual
tracking
%Column Headers in "data"
    %column1=no title
    %column2=Track Number aka, cell you are tracking
    %column3=slice number, i.e., 145 slices over the course of 24 hours=10
minute picture frames
    %column4=X pixel value
    %column5=Y pixel value
    %column6=distance (first value of each new track will be -1 since no
    %accumulated distance can be calculated)
    %column7=velocity (first value of each new track will be -1 since no
    %velocity from previous point can be calculated)
    %column8=pixel intensity

cellneighbor=load('WT_3_28_file2_left.txt'); %load the cell-neighbor text
file
%must be in increasing length and matrix dimensions must agree
%zeroes indicate no more neighbors left

%% Calculating Unit Vectors
num_slides=max(data(:,3));   %total number of pictures in movie
j=[1:num_slides:length(data)]; %beginning points of cells

k=[num_slides:num_slides:length(data)]; %ending points of cells
cells=data(:,2); %number of cells tracked

for cells=1:max(cells);
    initial_pts=data(j,4:5); %starting coordinates for each cell in
    %order
    ending_pts=data(k,4:5); %ending coordinates for each cell in
    %order
end

norm_coords=ending_pts-initial_pts; %normalizes each vector to start at
%the origin
x_norm_coords=norm_coords(:,1); %x coordinates of norm_coords
y_norm_coords=norm_coords(:,2); %y coordinates of norm_coords

for i=1:length(norm_coords);
    sqrdx(i)=x_norm_coords(i)^2;
    sqrdy(i)=y_norm_coords(i)^2;
    disp(i)=sqrt(sqrdx(i)+sqrdy(i)); %gives length of each vector

    unit_xr(i)=x_norm_coords(i)/disp(i); %x unit coordinate values
    unit_yr(i)=y_norm_coords(i)/disp(i); %y unit coordinate values
```

```
end

unit_x=unit_xr'; %converting row to column
unit_y=unit_yr'; %converting row to column

n=[1:1:length(unit_x)]';
unit_coords=[n unit_x unit_y]; %unit vector coordinates normalized to
%origin, what the correlation model calls when matching target to neighbor

clear i
%% Correlating Cell Direction-Target Cell and its Neighbors

for target=1:length(cellneighbor); %loop telling how many clusters
    %need to be correlated

    target_cell=1:length(cellneighbor);
        for i=1:length(cellneighbor);
            while target==target_cell(i)
                C=cellneighbor(target_cell(i),:);
%C calls each target cell and its neighbor in an array. Overwrites C each
time it passes through the while loop (after it completes the calculations
below).
              break
            end
        end
    clear i

    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unit_coords(m,2); %gives x coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unit_coords(m,3); %gives y coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"

            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end

    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
```

```
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
        %correlation is rewritten for each new cluster
    end

          cluster_avg_correlation=sum(correlation)/length(correlation);
          %average of of variable "correlation" for each target cell
          left_all_correlations(target)=cluster_avg_correlation;
          %variable listing averages for each target cell
          clear correlation C
end

left_sample_size=nnz(cellneighbor(:,2:end));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% RIGHT SIDE
clearvars -except left_all_correlations left_overall_correlation
clearvars -except left_stdev left_sterror left_all_correlations
left_sample_size

%% Loading Data
data=load('WT_3_28_file1_right.txt'); %original file from Image J manual
tracking
%Column Headers in "data"
    %column1=no title
    %column2=Track Number aka, cell you are tracking
    %column3=slice number
    %column4=X pixel value
    %column5=Y pixel value
    %column6=distance (first value of each new track will be -1 since no
    %accumulated distance can be calculated)
    %column7=velocity (first value of each new track will be -1 since no
    %velocity from previous point can be calculated)
    %column8=pixel intensity

cellneighbor=load('WT_3_28_file2_right.txt');
%load the cell-neighbor text file
%must be in increasing length and matrix dimensions must agree
%zeroes indicate no more neighbors left

%% Calculating Unit Vectors
num_slides=max(data(:,3));   %total number of pictures in movie
j=[1:num_slides:length(data)]; %beginning points of cells

k=[num_slides:num_slides:length(data)]; %ending points of cells
cells=data(:,2); %number of cells tracked

for cells=1:max(cells);
    initial_pts=data(j,4:5); %starting coordinates for each cell in
    %order
    ending_pts=data(k,4:5); %ending coordinates for each cell in
```

```
    %order
end

norm_coords=ending_pts-initial_pts; %normalizes each vector to start at
%the origin
x_norm_coords=norm_coords(:,1); %x coordinates of norm_coords
y_norm_coords=norm_coords(:,2); %y coordinates of norm_coords

for i=1:length(norm_coords);
    sqrdx(i)=x_norm_coords(i)^2;
    sqrdy(i)=y_norm_coords(i)^2;
    disp(i)=sqrt(sqrdx(i)+sqrdy(i)); %gives length of each vector

    unit_xr(i)=x_norm_coords(i)/disp(i); %x unit coordinate values
    unit_yr(i)=y_norm_coords(i)/disp(i); %y unit coordinate values
end

unit_x=unit_xr'; %converting row to column
unit_y=unit_yr'; %converting row to column


n=[1:1:length(unit_x)]';
unit_coords=[n unit_x unit_y]; %unit vector coordinates

clear i
%% Correlating Cell Direction-Target Cell and its Neighbors

for target=1:length(cellneighbor); %loop telling how many clusters need to be
correlated

    target_cell=1:length(cellneighbor);
        for i=1:length(cellneighbor);
            while target==target_cell(i)
                C=cellneighbor(target_cell(i),:);
%C calls each target cell and its neighbor in an array. Overwrites C
each time it passes through the while loop (after it completes the
calculations below).
              break
            end
        end
    clear i

    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unit_coords(m,2); %gives x coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unit_coords(m,3); %gives y coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
```

```
            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end
    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
    end

          cluster_avg_correlation=sum(correlation)/length(correlation);
          %average of of variable "correlation" for each target cell
          right_all_correlations(target)=cluster_avg_correlation;
          %variable listing averages for each target cell
          clear correlation C
end

right_sample_size=nnz(cellneighbor(:,2:end));

%% Calculating overall averages between left and ride sides
    %for one experiment, left side and right side
L_R_cluster_averages=[left_all_correlations right_all_correlations];
overall_correlation_average=mean(L_R_cluster_averages)
overall_correlation_stdev=std(L_R_cluster_averages)
overall_sterror=overall_correlation_stdev/sqrt(right_sample_size+left_sample_
size)

[status, msg_left]= xlswrite('overall_correlation_averages.xlsx',...
    L_R_cluster_averages,1);
%writes to excel file, gives status in command screen 1-success, 0-fail
```

Appendix B: Directional Model Using Euclidian Distances and Time Steps

Calculates one side of one experiment at a time

```
%% Correlating Cell-Neighbor Movement Using Time-Steps ~5  hrs
clear all
clc
%Created by J. Sanders on 8/25/2011
%last updated 5/5/2012

%% Loading Data
data=load('LTV_4_4_file1_right.txt'); %original file from Image J manual
%tracking
%Column Headers in "data"
    %column1=no title
    %column2=Track Number aka, Cell Number
    %column3=slice number, 145 slices over the course of 24 hours=10 minute
    %picture frames
    %column4=X pixel value
    %column5=Y pixel value
    %column6=distance (first value of each new track will be -1 since no
    %accumulated distance can be calculated)
    %column7=velocity (first value of each new track will be -1 since no
    %velocity from previous point can be calculated)
    %column8=pixel intensity

cellneighbor=load('LTV_4_4_file2_right.txt'); %load the cell-neighbor
%text file, must be in increasing length and matrix dimensions must agree
%Each file is seperated into left and right (unless otherwise notified)
%and loaded seperately

%% Splitting data files into time-steps
movie_length=max(data(:,3)); %total number of pictures in movie

j=[1:movie_length:length(data)]; %beginning points of each cell
l=[30:movie_length:length(data)]; %5 hr mark
m=[59:movie_length:length(data)]; %10 hr mark
n=[88:movie_length:length(data)]; %15 hr mark
o=[117:movie_length:length(data)]; %20 hr mark
k=[movie_length:movie_length:length(data)]; %ending points of each cell

cells=data(:,2); %number of cells tracked

 for cells=1:max(cells);
     begin=data(j,4:5); %beginning coordinates for each cell
     fivehrs=data(l,4:5); %5 hr mark coordinates for each cell
     tenhrs=data(m,4:5); %10 hr mark coordinates for each cell
     fifteenhrs=data(n,4:5); %15 hr mark coordinates for each cell
     twentyhrs=data(o,4:5); %20 hr mark coordinates for each cell
     ending=data(k,4:5); %end coordinates for each cell
 end

%% Calculating Unit Vector Distances For Time Steps
```

```
%beginning-5 hrs
begin_five=fivehrs(:,:)-begin(:,:); %normalizing the vector to origin
sqrd_begin_five=begin_five.^2;
sqrd_x_begin_five=sqrd_begin_five(:,1);
sqrd_y_begin_five=sqrd_begin_five(:,2);
for i=1:length(sqrd_x_begin_five);
    disp_begin_5(i)=sqrt(sqrd_x_begin_five(i)+sqrd_y_begin_five(i));
    unitx_begin_5(i)=begin_five(i,1)/disp_begin_5(i);
    unity_begin_5(i)=begin_five(i,2)/disp_begin_5(i);
end
unitmatrix_begin_5=[unitx_begin_5' unity_begin_5']; %defining unit
%vectors normalized from the origin from the beginning to 5 hr mark

%5 hrs-10 hrs
five_ten=tenhrs(:,:)-fivehrs(:,:); %normalizing the vector to the origin
sqrd_five_ten=five_ten.^2;
sqrd_x_five_ten=sqrd_five_ten(:,1);
sqrd_y_five_ten=sqrd_five_ten(:,2);
for i=1:length(sqrd_x_five_ten);
    disp_five_ten(i)=sqrt(sqrd_x_five_ten(i)+sqrd_y_five_ten(i));
    unitx_five_ten(i)=five_ten(i,1)/disp_five_ten(i);
    unity_five_ten(i)=five_ten(i,2)/disp_five_ten(i);
end
unitmatrix_five_ten=[unitx_five_ten' unity_five_ten']; %defining unit
%vectors normalized from the origin from the 5 hr to 10 hr mark

%10 hrs- 15 hrs
ten_fifteen=fifteenhrs(:,:)-tenhrs(:,:); %normalizing the vector to origin
sqrd_ten_fifteen=ten_fifteen.^2;
sqrd_x_ten_fifteen=sqrd_ten_fifteen(:,1);
sqrd_y_ten_fifteen=sqrd_ten_fifteen(:,2);
for i=1:length(sqrd_x_ten_fifteen);
    disp_ten_fifteen(i)=sqrt(sqrd_x_ten_fifteen(i)+sqrd_y_ten_fifteen(i));
    unitx_ten_fifteen(i)=ten_fifteen(i,1)/disp_ten_fifteen(i);
    unity_ten_fifteen(i)=ten_fifteen(i,2)/disp_ten_fifteen(i);
end
unitmatrix_ten_fifteen=[unitx_ten_fifteen' unity_ten_fifteen'];
%defining unit vectors normalized from the origin from the 10 hr to
%15 hr mark

%15 hrs-20 hrs
fifteen_twenty=twentyhrs(:,:)-fifteenhrs(:,:); %normalize vector to origin
sqrd_fifteen_twenty=fifteen_twenty.^2;
sqrd_x_fifteen_twenty=sqrd_fifteen_twenty(:,1);
sqrd_y_fifteen_twenty=sqrd_fifteen_twenty(:,2);
for i=1:length(sqrd_x_fifteen_twenty);

disp_fifteen_twenty(i)=sqrt(sqrd_x_fifteen_twenty(i)+sqrd_y_fifteen_twenty(i)
);
    unitx_fifteen_twenty(i)=fifteen_twenty(i,1)/disp_fifteen_twenty(i);
    unity_fifteen_twenty(i)=fifteen_twenty(i,2)/disp_fifteen_twenty(i);
end
unitmatrix_fifteen_twenty=[unitx_fifteen_twenty' unity_fifteen_twenty'];
%defining unit vectors normalized from the origin from the 15 hr to
%20 hr mark
```

```
%20 hrs-end
twenty_end=ending-twentyhrs(:,:); %normalizing the vector to the origin
sqrd_twenty_end=twenty_end.^2;
sqrd_x_twenty_end=sqrd_twenty_end(:,1);
sqrd_y_twenty_end=sqrd_twenty_end(:,2);
for i=1:length(sqrd_x_twenty_end);
    disp_twenty_end(i)=sqrt(sqrd_x_twenty_end(i)+sqrd_y_twenty_end(i));
    unitx_twenty_end(i)=twenty_end(i,1)/disp_twenty_end(i);
    unity_twenty_end(i)=twenty_end(i,2)/disp_twenty_end(i);
end
unitmatrix_twenty_end=[unitx_twenty_end' unity_twenty_end']; %defining
%unit vectors normalized from the origin from 20 hr mark to the end

%% Correlating from beginning to 5 hrs
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

    for i=1:length(cellneighbor);
    while target==target_cell(i)
        C=cellneighbor(target_cell(i),:);
        %C calls each target cell and its neighbor in an array.
        %Overwrites C  each time it passes through the
        %while loop (after it completes the calculations below).
       break
    end
    end
    clear i
    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unitmatrix_begin_5(m,1); %gives x coordinates of
target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unitmatrix_begin_5(m,2); %gives y coordinates of
target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"

            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end
    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
```

```
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)))
    end

average_correlation_begin_5hrs=sum(correlation)/length(correlation);
          %average of of variable "correlation" for each target cell
          all_correlations_begin_five(target)=average_correlation_begin_5hrs;
          %variable listing averages for each target cell
end
clear correlation p just_neighbors cluster_coords y_cluster x_cluster
clear y_array_value x_array_value i m C target target_cell

%% Correlating from 5-10 hrs

for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file

    target_cell=[1:length(cellneighbor)];

    for i=1:length(cellneighbor);
    while target==target_cell(i)
        C=cellneighbor(target_cell(i),:);
        %C calls each target cell and its neighbor in an array.
        %Overwrites C  each time it passes through the
        %while loop (after it completes the calculations below).
       break
    end
    end
    clear i
    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unitmatrix_five_ten(m,1); %gives x coordinates
of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unitmatrix_five_ten(m,2); %gives y coordinates
of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"

            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
```

```
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end
    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
    end
          average_correlation_five_ten=sum(correlation)/length(correlation);
          %average of of variable "correlation" for each target cell
          all_correlations_five_ten(target)=average_correlation_five_ten;
          %variable listing averages for each target cell
          clear C correlation
end
clear correlation p just_neighbors cluster_coords y_cluster x_cluster
clear y_array_value x_array_value i m C target target_cell

%% Correlating from 10-15 hrs

for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

    for i=1:length(cellneighbor);
    while target==target_cell(i)
        C=cellneighbor(target_cell(i),:);
        %C calls each target cell and its neighbor in an array.
        %Overwrites C  each time it passes through the
        %while loop (after it completes the calculations below).
       break
    end
    end
    clear i
    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unitmatrix_ten_fifteen(m,1); %gives x
coordinates
            %of target cell first then its subsequent neighbors as defined
            %by the variable "C"
            y_array_value(i)=unitmatrix_ten_fifteen(m,2); %gives y
coordinates
            %of target cell first then its subsequent neighbors as defined by
            %the variable "C"

            x_cluster=x_array_value'; %transposes into column vector
```

```
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end

    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
    end

average_correlation_ten_fifteen=sum(correlation)/length(correlation);
    %average of of variable "correlation" for each target cell

all_correlations_ten_fifteen(target)=average_correlation_ten_fifteen;
    %variable listing averages for each target cell
    clear correlation C
end
clear correlation p just_neighbors cluster_coords y_cluster
clear x_cluster y_array_value x_array_value i m C target target_cell

%% Correlating from 15-20 hrs

for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file

    target_cell=[1:length(cellneighbor)];

    for i=1:length(cellneighbor);
    while target==target_cell(i)
        C=cellneighbor(target_cell(i),:);
        %C calls each target cell and its neighbor in an array.
        %Overwrites C  each time it passes through the
        %while loop (after it completes the calculations below).
       break
    end
    end
    clear i
    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);
```

```
            x_array_value(i)=unitmatrix_fifteen_twenty(m,1); %gives x
coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unitmatrix_fifteen_twenty(m,2); %gives y
coordinates of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"

            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end
    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
    end

average_correlation_fifteen_twenty=sum(correlation)/length(correlation);
          %average of of variable "correlation" for each target cell

all_correlations_fifteen_twenty(target)=average_correlation_fifteen_twenty;
          %variable listing averages for each target cell
          clear correlation C
end
clear correlation p just_neighbors cluster_coords y_cluster
clear x_cluster y_array_value x_array_value i m C target target_cell

%% Correlating from 20-end hrs

for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file

    target_cell=[1:length(cellneighbor)];

    for i=1:length(cellneighbor);
    while target==target_cell(i)
        C=cellneighbor(target_cell(i),:);
        %C calls each target cell and its neighbor in an array.
        %Overwrites C  each time it passes through the
        %while loop (after it completes the calculations below).
       break
```

```
    end
    end
    clear i
    for i=1:nnz(C); %reads each row but only takes the non-zero length to
        %correlate target to each neighbor (has a "0" if no more neighbors
        %listed)
        for m=1:length(cellneighbor);
         while C(i)==target_cell(m);

            x_array_value(i)=unitmatrix_twenty_end(m,1); %gives x coordinates
of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"
            y_array_value(i)=unitmatrix_twenty_end(m,2); %gives y coordinates
of target
            %cell first then its subsequent neighbors as defined by the
            %variable "C"

            x_cluster=x_array_value'; %transposes into column vector
            y_cluster=y_array_value'; %transposes into column vector

            cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
            %variables x_cluster and y_cluster defined above
            just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
            %coordinates of just neighbors for each target cell
            break
         end
        end
    end
    %Scale of Correlation: 0-completely coorelated, 1-perpedicular 2-opposite
    %directions
        %the closer the answer is the 0, the more the pair move in the same
        %direction

    for p=1:length(just_neighbors); %calculates correlation values for
however
        %many neighbors a target has
        correlation(p)=1-dot(cluster_coords(1,:),(just_neighbors(p,:)));
    end

average_correlation_twenty_end=sum(correlation)/length(correlation);
          %average of variable "correlation" for each target cell
          all_correlations_twenty_end(target)=average_correlation_twenty_end;
          %variable listing averages for each target cell
          clear correlation C
end

%% Overall averages between each time step

all_correlations=[all_correlations_begin_five all_correlations_five_ten
all_correlations_ten_fifteen...
    all_correlations_fifteen_twenty all_correlations_twenty_end]
%outputs all correlation values

average_correlation=mean(all_correlations)
stdev_correlation=std(average_correlation)
```

```
%all_stdevs=[stdev1 stdev2 stdev3 stdev4 stdev5] %lists overall standard
%deviations in order of time steps

[status, msg]=xlswrite('timestep_correlation_averages.xls',all_correlations);
%writes to Excel file all average correlations for each target-neighbor cell
cluster tracked
```

Appendix C: Direction and Distance Model Using Time Steps

Calculates one side of one experiment at a time

```
%% Correlating Cell-Neighbor Movement Using Time-Steps ~5  hrs
clear all
clc
%Created by J. Sanders on 11/12/2011
%last updated 4/10/2012

%% Loading Data
data=load('MYO_4_3_file1_right.txt'); %original file from Image J manual
%tracking. Each file is seperated into left and right (unless otherwise
%notified) and loaded seperately
%Column Headers in "data"
    %column1=no title
    %column2=Track Number aka, Cell Number
    %column3=slice number, 145 slices over the course of 24 hours=10 minute
    %picture frames
    %column4=X pixel value
    %column5=Y pixel value
    %column6=distance (first value of each new track will be -1 since no
    %accumulated distance can be calculated)
    %column7=velocity (first value of each new track will be -1 since no
    %velocity from previous point can be calculated)
    %column8=pixel intensity

cellneighbor=load('MYO_4_3_file2_right.txt'); %load the cell-neighbor
%text file, must be in increasing length and matrix dimensions must agree
%Each file is seperated into left and right (unless otherwise notified)
%and loaded seperately

%% Splitting data files into time-steps
movie_length=max(data(:,3)); %total number of pictures in movie

j=[1:movie_length:length(data)]; %beginning points of each cell
l=[30:movie_length:length(data)]; %5 hr mark
m=[59:movie_length:length(data)]; %10 hr mark
n=[88:movie_length:length(data)]; %15 hr mark
o=[117:movie_length:length(data)]; %20 hr mark
k=[movie_length:movie_length:length(data)]; %ending points of each cell

cells=data(:,2); %number of cells tracked

 for cells=1:max(cells);
     begin=data(j,4:5); %beginning coordinates for each cell
     fivehrs=data(l,4:5); %5 hr mark coordinates for each cell
     tenhrs=data(m,4:5); %10 hr mark coordinates for each cell
     fifteenhrs=data(n,4:5); %15 hr mark coordinates for each cell
     twentyhrs=data(o,4:5); %20 hr mark coordinates for each cell
     ending=data(k,4:5); %end coordinates for each cell
 end

%% Calculating Unit Vector Distances For Time Steps
```

```
%beginning-5 hrs
begin_five=fivehrs(:,:)-begin(:,:); %normalizing the vector to the
%origin
sqrd_begin_five=begin_five.^2;
sqrd_x_begin_five=sqrd_begin_five(:,1);
sqrd_y_begin_five=sqrd_begin_five(:,2);
for i=1:length(sqrd_x_begin_five);
    dispr_begin_five(i)=sqrt(sqrd_x_begin_five(i)+sqrd_y_begin_five(i));
end
disp_begin_5=dispr_begin_five';

%5 hrs-10 hrs
five_ten=tenhrs(:,:)-fivehrs(:,:); %normalizing the vector to the
%origin
sqrd_five_ten=five_ten.^2;
sqrd_x_five_ten=sqrd_five_ten(:,1);
sqrd_y_five_ten=sqrd_five_ten(:,2);
for i=1:length(sqrd_x_five_ten);
    dispr_five_ten(i)=sqrt(sqrd_x_five_ten(i)+sqrd_y_five_ten(i));
end
disp_five_ten=dispr_five_ten';

%10 hrs- 15 hrs
ten_fifteen=fifteenhrs(:,:)-tenhrs(:,:); %normalizing the vector to
sqrd_ten_fifteen=ten_fifteen.^2;
sqrd_x_ten_fifteen=sqrd_ten_fifteen(:,1);
sqrd_y_ten_fifteen=sqrd_ten_fifteen(:,2);
for i=1:length(sqrd_x_ten_fifteen);
    dispr_ten_fifteen(i)=sqrt(sqrd_x_ten_fifteen(i)+sqrd_y_ten_fifteen(i));
end
disp_ten_fifteen=dispr_ten_fifteen';

%15 hrs-20 hrs
fifteen_twenty=twentyhrs(:,:)-fifteenhrs(:,:); %normalizing the vector
sqrd_fifteen_twenty=fifteen_twenty.^2;
sqrd_x_fifteen_twenty=sqrd_fifteen_twenty(:,1);
sqrd_y_fifteen_twenty=sqrd_fifteen_twenty(:,2);
for i=1:length(sqrd_x_fifteen_twenty);

dispr_fifteen_twenty(i)=sqrt(sqrd_x_fifteen_twenty(i)+sqrd_y_fifteen_twenty(i
));
end
disp_fifteen_twenty=dispr_fifteen_twenty';

%20 hrs-end
twenty_end=ending-twentyhrs(:,:); %normalizing the vector to the origin
sqrd_twenty_end=twenty_end.^2;
sqrd_x_twenty_end=sqrd_twenty_end(:,1);
sqrd_y_twenty_end=sqrd_twenty_end(:,2);
for i=1:length(sqrd_x_twenty_end);
    dispr_twenty_end(i)=sqrt(sqrd_x_twenty_end(i)+sqrd_y_twenty_end(i));
end
disp_twenty_end=dispr_twenty_end';

%% Correlating from beginning to 5 hrs
```

```
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

for i=1:length(cellneighbor);
while target==target_cell(i)
    C=cellneighbor(target_cell(i),:);
    %C calls each target cell and its neighbor in an array.
    %Overwrites C each time it passes through the
    %while loop (after it completes the calculations below)
   break
end
end
clear i
for i=1:nnz(C) %reads each row but only takes the non-zero length to
    %correlate target to each neighbor (has a "0" if no more neighbors
    %listed)
    for m=1:length(cellneighbor);
     while C(i)==target_cell(m)

        distance_valuer(i)=disp_begin_5(m,1); %matches distance of target and
        %its neighbors
        distance_value=distance_valuer'; %transposes into column vector

        x_array_value(i)=begin_five(m,1); %gives x coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"
        y_array_value(i)=begin_five(m,2); %gives y coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"

        x_cluster=x_array_value'; %transposes into column vector
        y_cluster=y_array_value'; %transposes into column vector

        cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
        %variables x_cluster and y_cluster defined above

        just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
        %coordinates of just neighbors for each target cell
        just_neighbors_distance=[distance_value(2:end)]; %gives distances
        %of just the neighbors
        break
     end
     end
     end

for p=1:length(just_neighbors); %calculates the euclidian distance gap
    %between the end of the target cell and end of the neighbor cell
    gap_distance(p)=(sqrt((((distance_value(1,1))^2+...
    (just_neighbors_distance(p,1))^2))-...
    (2*dot(cluster_coords(1,:),(just_neighbors(p,:))))))/distance_value(1,1);
end
      average_gap_distance=sum(gap_distance)/length(gap_distance); %average
      %of variable "correlation" for each target cell
```

```
      all_gap_distance_begin_five(target)=average_gap_distance; %variable
listing
      %averages for each target cell
 clear gap_distance C
end
clear correlation p just_neighbors cluster_coords y_cluster
clear x_cluster y_array_value x_array_value i m C target target_cell


%% Correlating from 5-10 hrs
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

for i=1:length(cellneighbor);
while target==target_cell(i)
    C=cellneighbor(target_cell(i),:);
    %C calls each target cell and its neighbor in an array.
    %Overwrites C each time it passes through the
    %while loop (after it completes the calculations below)
   break
end
end
clear i
for i=1:nnz(C) %reads each row but only takes the non-zero length to
    %correlate target to each neighbor (has a "0" if no more neighbors
    %listed)
    for m=1:length(cellneighbor);
     while C(i)==target_cell(m)

        distance_valuer(i)=disp_five_ten(m,1); %matches distance of target
and
        %its neighbors
        distance_value=distance_valuer'; %transposes into column vector

        x_array_value(i)=five_ten(m,1); %gives x coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"
        y_array_value(i)=five_ten(m,2); %gives y coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"

        x_cluster=x_array_value'; %transposes into column vector
        y_cluster=y_array_value'; %transposes into column vector

        cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
        %variables x_cluster and y_cluster defined above

        just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
        %coordinates of just neighbors for each target cell
        just_neighbors_distance=[distance_value(2:end)]; %gives distances
        %of just the neighbors
        break
     end
     end
```

```
     end

for p=1:length(just_neighbors); %calculates the euclidian distance gap
    %between the end of the target cell and end of the neighbor cell
    gap_distance(p)=(sqrt((((distance_value(1,1))^2+...
    (just_neighbors_distance(p,1))^2))-...
    (2*dot(cluster_coords(1,:),(just_neighbors(p,:))))))/distance_value(1,1);
end
      average_gap_distance=sum(gap_distance)/length(gap_distance); %average
      %of variable "correlation" for each target cell
      all_gap_distance_five_ten(target)=average_gap_distance; %variable
listing
      %averages for each target cell
 clear gap_distance C
end
clear correlation p just_neighbors cluster_coords y_cluster x_cluster
clear y_array_value x_array_value i m C targerget target_cell

%% Correlating from 10-15 hrs
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

for i=1:length(cellneighbor);
while target==target_cell(i)
    C=cellneighbor(target_cell(i),:);
    %C calls each target cell and its neighbor in an array.
    %Overwrites C each time it passes through the
    %while loop (after it completes the calculations below)
   break
end
end
clear i
for i=1:nnz(C) %reads each row but only takes the non-zero length to
    %correlate target to each neighbor (has a "0" if no more neighbors
    %listed)
    for m=1:length(cellneighbor);
     while C(i)==target_cell(m)

        distance_valuer(i)=disp_ten_fifteen(m,1); %matches distance of target
and
        %its neighbors
        distance_value=distance_valuer'; %transposes into column vector

        x_array_value(i)=ten_fifteen(m,1); %gives x coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"
        y_array_value(i)=ten_fifteen(m,2); %gives y coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"

        x_cluster=x_array_value'; %transposes into column vector
        y_cluster=y_array_value'; %transposes into column vector

        cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
```

```
        %variables x_cluster and y_cluster defined above

        just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
        %coordinates of just neighbors for each target cell
        just_neighbors_distance=[distance_value(2:end)]; %gives distances
        %of just the neighbors
        break
     end
     end
     end

for p=1:length(just_neighbors); %calculates the euclidian distance gap
    %between the end of the target cell and end of the neighbor cell
    gap_distance(p)=(sqrt((((distance_value(1,1))^2+...
    (just_neighbors_distance(p,1))^2))-...
    (2*dot(cluster_coords(1,:),(just_neighbors(p,:))))))/distance_value(1,1);
end
      average_gap_distance=sum(gap_distance)/length(gap_distance); %average
      %of variable "correlation" for each target cell
      all_gap_distance_ten_fifteen(target)=average_gap_distance; %variable
listing
      %averages for each target cell
 clear gap_distance C
end
clear correlation p just_neighbors cluster_coords y_cluster x_cluster
clear y_array_value x_array_value i m C target target_cell

%% Correlating from 15-20 hrs
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

for i=1:length(cellneighbor);
while target==target_cell(i)
    C=cellneighbor(target_cell(i),:);
    %C calls each target cell and its neighbor in an array.
    %Overwrites C each time it passes through the
    %while loop (after it completes the calculations below)
   break
end
end
clear i
for i=1:nnz(C) %reads each row but only takes the non-zero length to
    %correlate target to each neighbor (has a "0" if no more neighbors
    %listed)
    for m=1:length(cellneighbor);
     while C(i)==target_cell(m)

        distance_valuer(i)=disp_fifteen_twenty(m,1); %matches distance of
target and
        %its neighbors
        distance_value=distance_valuer'; %transposes into column vector

        x_array_value(i)=fifteen_twenty(m,1); %gives x coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"
```

```
        y_array_value(i)=fifteen_twenty(m,2); %gives y coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"

        x_cluster=x_array_value'; %transposes into column vector
        y_cluster=y_array_value'; %transposes into column vector

        cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
        %variables x_cluster and y_cluster defined above

        just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
        %coordinates of just neighbors for each target cell
        just_neighbors_distance=[distance_value(2:end)]; %gives distances
        %of just the neighbors
        break
     end
     end
     end

for p=1:length(just_neighbors); %calculates the euclidian distance gap
    %between the end of the target cell and end of the neighbor cell
    gap_distance(p)=(sqrt((((distance_value(1,1))^2+...
    (just_neighbors_distance(p,1))^2))-...
    (2*dot(cluster_coords(1,:),(just_neighbors(p,:))))))/distance_value(1,1);
end
      average_gap_distance=sum(gap_distance)/length(gap_distance); %average
      %of variable "correlation" for each target cell
      all_gap_distance_fifteen_twenty(target)=average_gap_distance; %variable
listing
      %averages for each target cell
 clear gap_distance C
end
clear correlation p just_neighbors cluster_coords y_cluster x_cluster
clear y_array_value x_array_value i m C target target_cell

%% Correlating from 20-end hrs
for target=1:length(cellneighbor); %length of cellneighbor=how many
    %target-neighbor cells were paired off in the cellneighbor data file
    target_cell=[1:length(cellneighbor)];

for i=1:length(cellneighbor);
while target==target_cell(i)
    C=cellneighbor(target_cell(i),:);
    %C calls each target cell and its neighbor in an array.
    %Overwrites C each time it passes through the
    %while loop (after it completes the calculations below)
   break
end
end
clear i
for i=1:nnz(C) %reads each row but only takes the non-zero length to
    %correlate target to each neighbor (has a "0" if no more neighbors
    %listed)
    for m=1:length(cellneighbor);
     while C(i)==target_cell(m)
```

```
        distance_valuer(i)=disp_twenty_end(m,1); %matches distance of target
and
        %its neighbors
        distance_value=distance_valuer'; %transposes into column vector

        x_array_value(i)=twenty_end(m,1); %gives x coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"
        y_array_value(i)=twenty_end(m,2); %gives y coordinates of target
        %cell first then its subsequent neighbors as defined by the
        %variable "C"

        x_cluster=x_array_value'; %transposes into column vector
        y_cluster=y_array_value'; %transposes into column vector

        cluster_coords=[x_cluster y_cluster]; %making a x,y matrix with
        %variables x_cluster and y_cluster defined above

        just_neighbors=[x_cluster(2:end) y_cluster(2:end)]; %gives the
        %coordinates of just neighbors for each target cell
        just_neighbors_distance=[distance_value(2:end)]; %gives distances
        %of just the neighbors
        break
     end
     end
     end

for p=1:length(just_neighbors); %calculates the euclidian distance gap
    %between the end of the target cell and end of the neighbor cell
    gap_distance(p)=(sqrt((((distance_value(1,1))^2+...
    (just_neighbors_distance(p,1))^2))-...
    (2*dot(cluster_coords(1,:),(just_neighbors(p,:))))))/distance_value(1,1);
end
      average_gap_distance=sum(gap_distance)/length(gap_distance); %average
      %of variable "correlation" for each target cell
      all_gap_distance_twenty_end(target)=average_gap_distance; %variable
listing
      %averages for each target cell
 clear gap_distance C
end

%% Overall averages between each time step
    %for one side experiment of one experiment only!
all_correlations=real([all_gap_distance_begin_five all_gap_distance_five_ten
all_gap_distance_ten_fifteen...
    all_gap_distance_fifteen_twenty all_gap_distance_twenty_end]);
%outputs all correlation values

average_correlation=mean(all_correlations)
all_stdevs=std(all_correlations)
```